Remarks/Arguments

With reference to the Office Action of December 12, 2006 Applicants offer the following remarks.

Art Rejections

1. Claims 1, 4, 7, and 10 – 35 USC §102, Altinel et al.

Claims 1, 4, 7, and 10 were rejected under 35 USC §102 as anticipated by Altinel and Franklin, "Efficient Filtering of XML Documents for Selective Dissemination of Information."

2. Claims 2, 3, 5, 6, 8, 9,and 11 – 35 USC §103(a) – Altinel and Vianu

Claims 2, 3, 5, 6, 8, 9, and 11 were rejected under 35 USC §103(a) as unpatentable over Altinel with Vianu.

3 Claims 12, 13, 14, 15, 16, 17, 18, 19, 20, 21 and 22 – 35 USC §103(a) – Gupta and Vianu

Claims 12, 13, 14, 15, 16, 17, 18, 19, 20, 21 and 22 were rejected under 35 USC §103(a) as unpatentable over Gupta with Vianu.

Art of Record

1. Altinel and Franklin, Efficient Filtering of XML Documents for Selective Dissemination of Information, states that information dissemination applications are gaining increasing popularity due to dramatic improvements in communications bandwidth and ubiquity. They next describe the problem that the sheer volume of data available drives the use of selective approaches to dissemination in order to avoid overwhelming users with unnecessary information. Existing mechanisms for selective dissemination typically rely on simple keyword matching or "bag of words" information retrieval techniques. The advent of XML as a standard for information exchange and the development of query languages for XML data enables the development of more sophisticated filtering mechanisms that take structure information into account. Altinel and Franklin describe several index organizations and search algorithms that they have developed for performing efficient filtering of XML documents for large-scale information dissemination systems. In the cited paper they describe these techniques and examine their performance across a range of document, workload, and scale scenarios.

Specifically cited is the passage at page 55, column 2,

> "In contrast, in SDI systems, large numbers of queries are stored, and the documents are individually matched to the queries. Thus, in an SDI system, it is necessary to *index the queries*. XFilter is unique in that it combines the scalable SDI approach of indexing queries with the ability to reference document structure (i.e., schema information) leading to scalable but precise filtering of documents for Internet-scale systems. "

and

> "The main structures used in the Filter Engine are depicted in Figure 3. Each XPath query is decomposed into a set of *path nodes* by the XPath parser. These path nodes represent the element nodes in the query and serve as the states of the FSM for the query. Path nodes are not generated for wildcard ("*") nodes."

And

> "These profiles are "standing queries", which are (conceptually) applied to all incoming documents."

Further citing

> "The other key inputs to an SDI system are the documents to be filtered."

And

> "When a document arrives at the Filter Engine, it is run through an XML Parser which then drives the process of checking for matching profiles in the Index. We use an XML

> parser that is based on the SAX interface, which is a standard interface for *event-based* XML parsing [Meg98]. We developed the parser using the *expat* toolkit [Cla99a], which is a non-validating XML processor. "

were used to show anticipation of original claim 1.

2. Yannis Papakonstantinou andVictor Vianu, DTD Inference for Views of XML Data, describes the inference of Data Type Definitions DTDs for views of XML data, using an abstraction that focuses on document content structure. The views are defined by a query language that produces a list of documents selected from one or more input sources. The selection conditions involve vertical and horizontal navigation, thus querying explicitly the order present in input documents. As background, Vianu et al. point to several strong limitations in the descriptive ability of current DTDs and the need for extending them with (i) a sub typing mechanism and (ii) a more powerful specification mechanism than regular languages, such as context-free languages. With these extensions, Vianu et al. show that one can always infer tight DTDs, that precisely characterize a selection view on sources satisfying given DTDs. They also show important special cases where one can infer a tight DTD without requiring extension.

U.S. Published Patent Application 20040034651-A1 to Gupta et al. for Data Source Interation System and Method describes a method and program product for integrating different data sources. Gupta describes steps of obtaining semantic information from each of these different data sources, creating a conceptual model of the data source using the semantic information, and accessing one or more secondary knowledge sources. The secondary information sources contain information regarding the relations of data from different of the databases, so that an integrated semantic model of all of the databases may be created. Queries can then be processed using the integrated semantic model.

The following passage in Gupta is relied upon in the Office Action:

"[0266] Formally, the object-oriented class structure of M is given by a mapping .PHI_M: DTD(M).=>.SIGMA.(M). Technically, PHI_M. is straightforward to implement as it amounts to a simple syntactic transformation from XML elements to F-logic expressions (e.g., using an XML parser whose output is "pretty-printed" to F-logic, or using the XML stylesheet/transformation language XSL(T)). The difficulty consists in choosing the most appropriate "semantically adequate" representation in F-logic of the underlying, XML-encoded, object model."

Discussion

Status of the Claims.

Claims 1-22 were originally presented for Examination. All of the claims were rejected in the Office Action of December 12, 2006. Applicants have amended their claims to particularly point out and describe their invention, and distinguish over the art of record.

Table – Claim Status

Claim	Status
1	Amended
2	Canceled, limitations placed in Claim 1
3	Dependent on claim 1
4	Amended to be parallel to claim 1
5	Canceled
6	Original
7	Previously Canceled
8	Previously Canceled
9	Previously Canceled
10	Amended to be parallel to claim 1
11	Canceled., limitations placed in claim 10
12	Previously Canceled
13	Previously Canceled
14	Amended to be parallel to claim 1

Claim	Status
15	Original
16	Previously Canceled
17	Previously Canceled
18	Amended to be parallel to claim 1
19	Canceled, limitations placed in claim 19
20	Amended to be parallel to claim 1
21	Canceled, limitations placed in claim 20
22	Amended to be parallel to claim 1

This is a total reduction of 12 claims, i.e., a reduction of fifty four percent (54%) in the number of claims.

Exemplary Claim

Claim 1, as amended, is exemplary.

1. (Currently Amended) A document-searching system for searching a document having a hierarchical structure with elements separated by element identifiers, comprising:

a compiling device for generating a query automaton by storing an input query expression, performing parsing, identifying different types of nodes in said element identifiers, wherein the compiling device generates and registers a state transition by replacing an axis including an axis in the opposite direction and a logical expression including a conjunction or a negative expression while keeping an input query expression equal in terms of search, wherein the compiling device generates a query automaton including a plurality of states of the backward nodes, a condition for transition, and at least a search state;

a query automaton storage device for storing the query automaton generated by said compiling device;

a query automaton evaluator for reading out said query automaton from said storage device and storing said automaton, while reading in said document and performing a stream search by using states of a plurality of different types of nodes in said element identifiers included in said document and said query automaton, said query automaton evaluator determining a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node, and outputting the searched node and

a search result storage means for storing the output of the query automaton evaluator, and for thereafter outputting the stored output of the query automaton evaluator and the output of the searched node.

Discussion of patentability will be with respect to claim 1, since the newly added limitations are common to all of independent claims.

Discussion: Art Rejection

The overarching issue is whether the claims, as limited by the newly added clauses and limitations are allowable over the art of record.

All of the claims either contain one of the two above claim limitations or are dependent on base claims which contain one of the above limitations.

Claim 1 will be analyzed in detail. Amended claim 1 recites:

> *A document-searching system for searching a document having a hierarchical structure with elements separated by element identifiers, comprising:*
>
> *a compiling device for generating a query automaton by storing an input query expression, performing parsing, identifying different types of nodes in said element identifiers, wherein the compiling device generates and registers a state transition by replacing an axis including an axis in the opposite direction and a logical expression including a conjunction or a negative expression while keeping an input query expression equal in terms of search, wherein the compiling device generates a query automaton including a plurality of states of the backward nodes, a condition for transition, and at least a search state;*
>
> *a query automaton storage device for storing the query automaton generated by said compiling device;*
>
> *a query automaton evaluator for reading out said query automaton from said storage device and storing said automaton, while reading in said document and performing a stream search by using states of a plurality of different types of nodes in said element identifiers included in said document and said query automaton, said query automaton evaluator determining a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node, and outputting the searched node and*
>
> *a search result storage means for storing the output of the query automaton evaluator, and for thereafter outputting the stored output of the query automaton evaluator and the output of the searched node.*

Analyzing the claim in detail, the claim requires:

A document-searching system
a hierarchical structure
elements separated by element identifiers,

a compiling device
for generating a query automaton
by storing an input query expression,
performing parsing,
identifying different types of nodes in said element identifiers,
wherein the compiling device generates and registers a state transition by replacing an axis including an axis in the opposite direction and a logical expression including
a conjunction or a negative expression
while keeping an input query expression equal in terms of search,
wherein the compiling device generates a query automaton including
a plurality of states of the backward nodes,
a condition for transition, and
at least a search state;

a query automaton storage device
for storing the query automaton generated by said compiling device;

a query automaton evaluator for
reading out said query automaton from said storage device and
storing said automaton,
while reading in said document and performing a stream search by
using states of a plurality of different types of nodes in said element identifiers included in said document and said query automaton,
said query automaton evaluator determining a state transition of a node under determination by
storing a left node and a lower node
in correspondence with an identified element identifier, and
evaluating said query automaton with a search result of said left node and said lower node, and
outputting the searched node and

a search result storage means
storing the output of the query automaton evaluator, and
outputting
the stored output of the query automaton evaluator and
the output of the searched node.

In order to be an "anticipation" of the claimed invention, a reference must show each and every element of the claim. Altinel et al. fail this standard. The reference does not teach or even suggest the newly added claim limitations of

> *wherein the compiling device generates and registers a state transition by replacing an axis including an axis in the opposite direction and a logical expression including a conjunction or a negative expression while keeping an input query expression equal in terms of search, wherein the compiling device generates a query automaton including a plurality of states of the backward nodes, a condition for transition, and at least a search state*

and

> *said query automaton evaluator determining a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node,*

Altinel et al fail this test, reciting:

> In contrast, in SDI systems, large numbers of queries are stored, and the documents are individually matched to the queries. Thus, in an SDI system, it is necessary to *index the queries*. XFilter is unique in that it combines the scalable SDI approach of indexing queries with the ability to reference document structure (i.e., schema information) leading to scalable but precise filtering of documents for Internet-scale systems.

and

> Each XPath query is decomposed into a set of *path nodes* by the XPath parser. These path nodes represent the element nodes in the query and serve as the states of the FSM for the query. Path nodes are not generated for wildcard ("*") nodes. A path node contains the following information:

And

> The process of filtering and delivering documents based on user interests is sometimes referred to as Selective Dissemination of Information (SDI). Figure 1 shows a generic architecture for an XML-based SDI system. There are two main sets of inputs to the system: user profiles and data items (i.e., documents). User profiles describe the information preferences of individual users. In most systems these profiles are created by the users, typically by clicking on items in a Graphical User Interface. In some systems, however, these profiles can be learned automatically by the system through the application of machine learning techniques to user access traces. The user profiles are converted into a format that can be efficiently stored and evaluated by the Filter Engine. These profiles are "standing queries", which are (conceptually) applied to all incoming

documents.

These passages do not anticipate the claimed invention.

Vianu et al discloses a view definition language that queries labeled ordered trees. Even with Altinel, the addition of a view definition language does not teach or suggest Applicants' claimed invention.

Gupta's disclosure of

> "[0266] Formally, the object-oriented class structure of M is given by a mapping .PHI_M: DTD(M).=>.SIGMA.(M). Technically, PHI_M. is straightforward to implement as it amounts to a simple syntactic transformation from XML elements to F-logic expressions (e.g., using an XML parser whose output is "pretty-printed" to F-logic, or using the XML stylesheet/transformation language XSL(T)). The difficulty consists in choosing the most appropriate "semantically adequate" representation in F-logic of the underlying, XML-encoded, object model."

does not supplement the deficiencies of Altinel and Vianu.

Altinel et al., either alone or taken with one or both of Vianu and/or Gupta, does not teach or suggest Applicants'claimed invention of—

> *A document-searching system for searching a document having a hierarchical structure with elements separated by element identifiers, comprising:*
>
> *a compiling device for generating a query automaton by storing an input query expression, performing parsing, identifying different types of nodes in said element identifiers, wherein the compiling device generates and registers a state transition by replacing an axis including an axis in the opposite direction and a logical expression including a conjunction or a negative expression while keeping an input query expression equal in terms of search, wherein the compiling device generates a query automaton including a plurality of states of the backward nodes, a condition for transition, and at least a search state;*
>
> *a query automaton storage device for storing the query automaton generated by said compiling device;*
>
> *a query automaton evaluator for reading out said query automaton from said storage device and storing said automaton, while reading in said document and performing a stream search by using states of a plurality of different types of nodes in said element identifiers included in said document and said query automaton, said query automaton evaluator determining a state transition of a node under determination by storing a left*

node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node, and outputting the searched node and

a search result storage means for storing the output of the query automaton evaluator, and for thereafter outputting the stored output of the query automaton evaluator and the output of the searched node.

With the further limitations of

wherein the compiling device generates and registers a state transition by replacing an axis including an axis in the opposite direction and a logical expression including a conjunction or a negative expression while keeping an input query expression equal in terms of search, wherein the compiling device generates a query automaton including a plurality of states of the backward nodes, a condition for transition, and at least a search state

and

said query automaton evaluator determining a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node,

Conclusion

Based on the above discussion, it is respectfully submitted that the pending claims describe an invention that is statutory subject matter and is properly allowable to the Applicants.

If any issues remain unresolved despite the present amendment, the Examiner is requested to telephone Applicants' Attorney at the telephone number shown below to arrange for a telephonic interview before issuing another Office Action.

Applicants would like to take this opportunity to thank the Examiner for a thorough and competent examination and for courtesies extended to Applicants' Attorney.